AEC RESEARCH AND
DEVELOPMENT REPORT

NEW YORK UNIVERSITY
INSTITUTE OF MATHEMATICAL SCIENCES
LIBRARY
25 Waverly Place, New York 3, N. Y.

NYO 6482

NEW YORK UNIVERSITY

institute of mathematical sciences

AEC Computing Facility

New York, New York

NEW YORK UNIVERSITY
INSTITUTE OF MATHEMATICAL SCIENCES
LIBRARY
25 Waverly Place, New York 3, N. Y.

NYO-6482
Copy 1

A REPORT ON A MODIFIED ROUTH METHOD FOR LOCATING ROOTS OF POLYNOMIALS

by

Eugene Isaacson
Michael Held
Thomas Seidman

AEC Computing Facility
Institute of Mathematical Sciences
New York University

CONTRACT NO. AT(30-1)-1480

December 1, 1955

NEW YORK UNIVERSITY
INSTITUTE OF MATHEMATICAL SCIENCES
LIBRARY
25 Waverly Place, New York 3, N. Y.

Table of Contents

Section		Page
1	Introduction	3
2	Mathematical Background	6
3	Machine Aspects	12
Appendix I	Operating Instructions	20
Appendix II	Sample Problem	22

Abstract

A method for obtaining the complex roots of polynomials of degree less than or equal to twenty with real coefficients is described. Both the mathematical and Univac aspects are discussed.

Section 1. Introduction

A procedure[1] for obtaining the real and complex roots of polynomials of degree twenty or less with real coefficients has been tested on the Univac. In general the program produces answers correct to six or seven significant figures (on the basis of floating point operations with nine-digit numbers). Some outside intervention is necessary when there are roots which are "very close" to each other, or when the coefficients of the polynomial have "too wide" a spread (see Section 3 of this report).

The procedure makes repeated use of the Euclidean algorithm for finding the greatest common divisor of two polynomials. The fact that the Cauchy formula which gives the number of zeros to the "left" of a straight line can be evaluated by performing the Euclidean algorithm is derived in Section 2.

It was found to be a matter of mathematical necessity to first eliminate the occurrence of multiple

1. The method used has been described in the literature by T. C. Fry (Quarterly of Applied Mathematics, Vol. III, No. 2, July 1945) who states that G. R. Stibitz has programmed such a procedure for polynomials of degree ten or less. The method may ultimately be traced back to Routh.

roots. This is done by factoring out the greatest common divisor of the polynomial and its derivative. (Since the principal tool used in the location of the roots is based on the Euclidean algorithm for finding the greatest common divisor of two polynomials, this is a simple precaution.)

The next step is to determine a circle about the origin in which the roots lie. A simple estimate of the form

$$|z| \leq \frac{\max|a_i|}{|a_n|} + 1 + 10^{-6} = B$$

is used, where

$$P_n(z) = a_n z^n + a_{n-1} z^{n-1} + \cdots + a_1 z + a_0 .$$

The abscissae of the roots are now located by successive subdivision as follows. A record of the lower and upper bounds of the abscissae of each of the n roots $z_1, z_2, \ldots, z_n$ is kept; for example, at the start

$$-B \leq R_e z_i \leq +B \text{ for } i = 1, 2, \ldots, n.$$

If the upper bound G_1 and the lower bound g_1 of the abscissa of z_1 satisfy the inequality

$$\frac{G-g}{|G+g|+1} < 10^{-8}$$

then the average $\frac{1}{2}(g_1+G_1)$ is taken and recorded as the abscissa of z_1. The upper and lower bounds of z_2 are next examined, etc. (We have assumed that $R_e z_1 \leq R_e z_2 \leq \cdots \leq R_e z_{n-1} \leq R_e z_n$).

Suppose that z_K is the first root for which the upper, G_K, and lower, g_K, estimates of the abscissa fail to satisfy the inequality ⊙ above. The program explained in Section 2 then finds the number of roots to the left of the line

$$z = \tfrac{1}{2}(g_K+G_K) + it, \quad -\infty < t < +\infty.$$

Call this number η. If $\frac{1}{2}(g_K+G_K)$ is less than $G_\nu (\nu=K, K+1, \ldots, \eta)$, it replaces G_ν. If $\frac{1}{2}(g_K+G_K)$ is greater than $g_\nu (\nu=\eta+1, \eta+2, \ldots, n)$ it replaces g_ν. This process is repeated until g_K and G_K satisfy inequality ⊙ above. The estimates of the real part of z_{K+1} are then examined, etc.

In this way the abscissae of the roots $z_1, \ldots, z_n$ are determined (usually to within seven significant figures).

The ordinates of the roots are now found by examining the polynomials $U^{(K)}(t)$ and $V^{(K)}(t)$ given as the real and imaginary parts of $P_n(R_e z_K+it) = U^{(K)}(t) + iV^{(K)}(t)$. It is easy to see that $U^{(K)}(t)$ and

$V^{(K)}(t)$ have $Im z_K$ as a common real root. Unfortunately, $U^{(K)}(t)$ and $V^{(K)}(t)$ will have as common roots all pairs of roots of $P_n(z) = 0$ which are symmetrically situated with respect to the line $z = R_e z_K + it$. Therefore, to find the ordinate of the root z_K, the greatest common divisor $Q^{(k)}$ of $U^{(K)}$ and $V^{(K)}$ is found. The real roots of $Q^{(K)}$ are found by determining the abscissae of its roots and recording those whose imaginary parts vanish. ($Q^{(K)}$ will be either odd or even if $P_n(z)$ has real coefficients and it is possible to halve the degree of $Q^{(K)}$ through a simple change of variable.)

Section 2. Mathematical Background

In order to locate the zeros of the polynomial $P_n(z)$, we first determine a large circle C in which they lie. A simple bound is given by $|z| \leq \frac{\max|a_i|}{|a_n|} + 1 + 10^{-6}$ where $P_n(z) = a_n z^n + \cdots + a_1 z + a_0$. Application is then made of the Cauchy formula, which counts the number of zeros in the interior of a simple Jordan curve, in a process of successive subdivision of the circumscribed square around C. In this way, the coordinates of the zeros may be determined theoretically to any desired degree of accuracy.

The Cauchy formula

$$(1) \qquad N = \frac{1}{2\pi i} \oint_B d[\log P_n(z)]$$

gives the number of roots of $P_n(z) = 0$ contained within the curve B (if no roots are on B). It would be possible in principle to perform a numerical integration to evaluate the number N. But, by modifying B in (1) so that we obtain a formula for the number of zeros to the "left" of any straight line, it becomes possible to evaluate the resulting integral by simpler algebraic processes. We now give the details of the above program.

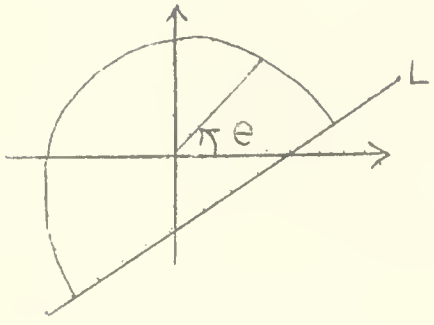
Formula (1) counts the total number of revolutions about the origin made by the radius vector drawn from the origin to the point $P_n(z)$ as z varies around B. Let us assume B consists of an arc of a circle of large radius centered at the origin, together with its chord. Suppose that the chord lies on a fixed straight line L and that the circular arc lies to the "left" of the line L. If we let the radius of the circular arc increase, the formula (1) will yield the number of zeros, q, to the left of the line L. The change in angle, $\Delta\Phi$, of $P_n(z) = Re^{i\Phi}$ over the circular arc will tend to $n\pi$ as the radius increases since for $z = re^{i\theta}$ and r large

$$P_n(z) \doteq a_n z^n + a_{n-1} z^{n-1} + \cdots + a_1 z + a_0$$

$$P_n(z) = a_n z^n \left(1 + \frac{a_{n-1}}{a_n z} + \cdots + \frac{a_0}{a_n z^n}\right)$$

$$\approx a_n z^n = a_n r^n e^{in\theta}$$

and $\Delta\theta \longrightarrow \pi$ as $r \longrightarrow \infty$.



Therefore

$$q = \frac{n}{2} + \frac{1}{2\pi i}\oint_L d \log P_n(z) .$$

The principle objective of the present section is to show how the change $\Delta\mathbb{D}$ in the angle $\mathbb{D}$ as z varies over the line L may be expressed algebraically. To this end we represent the points on the line L by $z = \gamma + t\delta$, where γ and δ are fixed complex numbers and t varies from $-\infty$ to $+\infty$. Then

$$P_n(z) = P_n(\gamma+t\delta) = U(t) + iV(t) = R(t)e^{i\mathbb{D}(t)}.$$

(We may assume that the polynomial V(t) is of degree $\geq$ the degree of U(t), otherwise we would consider $iP_n(z)$ which has the same zeros as $P_n(z)$.) The number

of revolutions made by $\mathbb{D}(t)$ can be found by appropriately counting the zeros of $V(t)$. Since the degree of V is not zero and is $\geq$ the degree of U, we know that $\mathbb{D}(t)$ as $t \longrightarrow \pm \infty$ tends to an angle which is not 0 or π. This means that if we count the number of "positive" crossings of the positive and negative real axis made by the radius vector of angle $\mathbb{D}(t)$ and subtract the number of "negative" crossings, we will obtain twice the number of revolutions made by the radius vector. We now describe how the counting may be performed algebraically.

A "positive" crossing of the real axis in the (U,V) plane is one for which V increases from negative to positive values if U is positive; while if U is negative V decreases from positive to negative values as t increases. The "negative" crossings are those in which V has the opposite behavior.

Now, in order to count the number of crossings; we first use the Euclidean algorithm for finding the greatest common divisor of V and U in the form

$$V = Uq_1 - r_1$$
$$U = r_1 q_2 - r_2$$
$$r_1 = r_2 q_3 - r_3$$
$$\vdots$$

$$r_{K-2} = r_{K-1}q_K - r_K$$

$$r_{K-1} = r_K q_{K+1}$$

where degree of $U \leq$ degree of V

degree of $r_1 <$ degree of U

degree of $r_{j+1} <$ degree of r_j

If we now consider the number $M(t)$ of changes of sign in the sequence of terms $(V, U, r_1, r_2, \ldots r_K)$, we observe that $M(t)$ can change as t increases only when V changes sign since

(1) V and U have no common zero (by the assumption that the line L was zero free)

(2) Whence no two successive terms vanish simultaneously.

(3) Hence the nearest neighbors of a term which vanishes have opposite sign.

(4) Thence the only way for M to change is by having V change sign, since r_K cannot vanish.

Now the number $M(t)$ decreases for a positive crossing and increases for a negative crossing.

Hence

$$M(-\infty) - M(+\infty) = 2 \cdot \text{number of revolutions of } \mathbb{D} \text{ over L}$$

whence

$$q = \frac{n}{2} + \frac{M(-\infty) - M(+\infty)}{2} \quad .$$

We note that only the signs and the degrees of the leading coefficients of $(V, U, r_1, \ldots, r_K)$ are needed to determine the numbers $M(-\infty)$ and $M(+\infty)$. (In the event that $P_n(z)$ has real coefficients, and the line L is a vertical line, one of the functions V,U will be odd and the other will be even. The sequence $(V, U, r_1, \ldots, r_K)$ will therefore alternately be odd and even functions. It is therefore possible to compute $M(-\infty) - M(+\infty)$ from a knowledge only of the signs of the leading coefficients of $(V, U, r_1, \ldots, r_K)$. For example, if all of the leading coefficients have positive signs, then $M(-\infty) = k + 1$ while $M(+\infty) = 0$. In other words if the leading coefficients of two neighboring terms have the same sign, they contribute + 1 to $M(-\infty)$ and 0 to $M(+\infty)$ and vice-versa if the adjacent leading coefficients have opposite sign they contribute 0 to $M(-\infty)$ and + 1 to $M(+\infty)$. It is therefore possible to compute $M(-\infty) - M(+\infty)$ without storing all of the coefficients of all of the terms $(V, U, r_1, \ldots, r_K)$ in the process of performing the greatest common divisor algorithm. That is, a single intermediate counter can be used to store $M_\nu(-\infty) - M_\nu(+\infty)$, the difference in the number of sign changes of the sequence $(V, U, r_1, \ldots, r_\nu)$, while the term $r_{\nu+1}$ is found from the knowledge of the two preceding terms $r_{\nu-1}$ and r_ν.

After r_{y+1} is found, r_{y-1} may be erased. The signs of the leading coefficient of r_{y+1} and r_y can be compared to find the number $M_{y+1}(-\infty) - M_{y+1}(+\infty)$ which represents the difference in the number of sign changes in the sequence $(V, U, r_1, \ldots, r_y, r_{y+1})$ etc.)

Section 3. Machine Aspects

In applying the method discussed in the preceeding sections on the Univac there are certain difficulties, some of which can only partially be overcome.

Consider first the sequence $(V, U, r_1, r_2, \ldots, r_K)$ and the Euclidean algorithm which leads to it

$$V = q_1 U - r_1$$
$$U = q_2 r_1 - r_2$$
$$r_1 = q_3 r_2 - r_3$$
$$\vdots$$
$$r_{K-2} = q_K r_{K-1} - r_K$$
$$r_{K-1} = q_{K+1} r_K \quad .$$

The degree of V is known, it is the same as the degree of the original polynomial. However, in order to successfully carry out the algorithm, one must be able to determine correctly the degree of U and of the r's. This requires a knowledge of what terms are to be considered as being

zero. Further, it is important to know what is to be considered as zero so as to know when to terminate the algorithm. In an effort to circumvent this difficulty an artifice was devised. Observe first that each member of the sequence $(V, U, r_1, \ldots, r_K)$ is a polynomial which contains exclusively odd or even degree terms. We then assume that the degree of U is one less than the degree of V. If in performing the necessary division this assumption leads to an exponent overflow or a zero divisor this fact is noted on the supervisory control typewriter (SCT) and the division is performed again -- this time assuming that the degree of U is 3 less than the degree of V. This process is repeated until the division is successfully completed. An entirely similar procedure is followed throughout the algorithm. The algorithm is allowed to continue until the final term of the sequence r_K is a constant (degree zero polynomial). The reason for this is clear. The final term of the sequence should be the greatest common divisor (g.c.d) of V and U. This g.c.d. will be a constant provided the vertical line we are trying does not pass through a root exactly and provided there are no roots symmetrically placed on either side of this line. These assumptions are tacitly made.

By successive applications thereof (see section 1) we can hope to find the abscissas of the roots. There are, however, two things which may prevent this.

(1) We may try a vertical line and get the result that there are η roots to the left of that line. But from previous computations we may have already "found" more than η roots to the left of that line. This inconsistency stops the Univac and the type out "ηN error" occurs. There is nothing more one can do in this situation.

(2) The exact opposite can occur. We may find that there are η roots to the left of a vertical line when in reality there are less than η roots there. No check can be made for this error, and, in general if it occurs, it will lead to poor accuracy in the final result.

These two errors occur only in the neighborhood of a root and particularly in the neighborhood of a multiple root. In an effort to keep the occurrence of these errors down to a minimum multiple roots are eliminated at the very outset of the computation by factoring out the g.c.d. of the polynomial and its derivative. An attempt is made to compute the proper multiplicity of each root (along with an error estimate) at the end of the computation (see below).

Once the abscissas have been found, one can proceed

in two different ways in order to obtain the ordinates. The basic method (Sections 1 and 2) can be used so as to yield the number of roots above or below any given horizontal line. This, however, would introduce complex arithmetic and lead to extraneous roots. Therefore a different method was adopted. Let $P_n(z)$ be the polynomial that we are interested in, and assume that we already know that α is an abscissa and that we wish to find a corresponding ordinate β. Let $z = \alpha + it$ where t is a variable. Then $P_n(z) = P_n(\alpha+it) = U(t) + iV(t)$. Then clearly we must have $U(\beta) = V(\beta) = 0$. The converse is also true; i.e., if $U(\beta) = V(\beta) = 0$ for real β then β is an ordinate corresponding to the abscissa α. It is obvious that the real roots of $g(t)$, the g.c.d. of $U(t)$ and $V(t)$, are the ordinates that we desire. We notice first that this g.c.d. $g(t)$ contains only odd or only even degree terms. If it contains only odd degree terms then a factor t can be removed and thus we may assume that $g(t)$ has only even degree terms. A simple change of variable then enables us to halve the degree of the g.c.d. We must then find the real roots of this "reduced g.c.d.", $g_r(t)$. If the degree of $g_r(t)$ is one, the results can be read right off. If the degree is two, a simple application of the quadratic formula gives the desired result. If the degree is greater

than or equal to three the typeout "G.C.D. DEG 6+" occurs. Then using $g_r(t)$ and the original basic method (Sections 1 and 2) we find the abscissas of $g_r(t)$. An ε - test against the remainder after synthetic division is then used to find out which of these abscissas are real roots. It is possible (although unlikely) for this test to go astray. If this happens the Univac stops and the typeout "extra roots" occurs at a later time.

The main difficulty is in finding the g.c.d. of V(t) and U(t). The Euclidean algorithm must be stopped at precisely the right step -- not too soon and not too late. That is to say we must have a precise way of telling when a division yields a remainder which is to be considered as being zero. The only check available to us at this point is that we know the g.c.d. should not be a constant. If it is calculated to be a constant then the ε used to test when a division yields a remainder of zero is multiplied by ten and the g.c.d. is re-calculated. This process is continued until either a non-constant g.c.d. is obtained or until the original ε has been multiplied by 10^{10}, at which time the Univac stops and the typeout "g.c.d. const." occurs. The g.c.d. finally obtained still may not be the correct one. If this happens either the Univac will stop and type out "extra roots" at a later time or the error will manifest

itself in bad error estimates (see below).

After all the roots of the reduced (no multiple roots) polynomial have been found, we try to obtain an error estimate and the proper multiplicity for each root. We proceed as follows: Let $P_n(z)$ be the original unreduced (with multiple roots possible) polynomial, and let a + ib be a root. Then make the substitution $z = a + it$:

$$P_n(z) = P_n(a+it) = U(t) + iV(t).$$

Divide U(t) synthetically by (t-b) and get a remainder $r_u^{(0)}$ and a quotient $q_u^{(0)}$. Divide V(t) synthetically by (t-b) and get a remainder $r_v^{(0)}$ and a quotient $q_v^{(0)}$. Then $\left\{|r_u^{(0)}| + |r_v^{(0)}|\right\}/a_n$ where a_n is the leading coefficient of $P_n(z)$ is the error estimate used for the root a + ib.

Now divide both $q_u^{(0)}$ and $q_v^{(0)}$ synthetically by (t-b) and get remainders $r_u^{(1)}$, $r_v^{(1)}$ and quotients $q_u^{(1)}$, $q_v^{(1)}$ respectively. If <u>both</u> $r_u^{(1)}$ and $r_v^{(1)}$ are less than some computed ε, then a + ib is said to have multiplicity two. By repeated such synthetic divisions we may hope to compute the proper multiplicity of each root.

As a matter of actual practice things do not always work out quite so nicely, and the sum total of the multiplicities may not add up to the degree of the original

polynomial. Particularly in the case of fairly high multiplicities (say > 3) the indicated multiplicity may not be correct (generally it will be less than the true one). A multiplicity of zero indicates that the computed root is so far from the true root that it should not be considered as a root at all.

Many of the machine difficulties mentioned in the preceeding paragraphs can be completely or partially avoided by scaling. For example the existing routine will not solve properly the polynomial $P_n(z) = 10^8 z^8 + 10^7 z^7 + 10^6 z^6 + \cdots + 10z + 1$; however, no trouble is encountered with the polynomial $P_n(z) = z^8 + z^7 + \cdots + z + 1$. The optimum condition seems to be one in which all the coefficients are of the same order of magnitude, and, the closer one can come to this condition, by scaling if necessary, the better will be the results obtained.[1]

The time taken to complete a problem is a function of the degree of the polynomial and of the geometry of its roots. The twentieth roots of unity take slightly

1. Note: If the polynomial being solved is given by $P_n(z) = a_n z^n + \cdots + a_1 z + a_0$, then a necessary condition for the operation of the routine is that $(\max |a_i|)^n$ divided by $(a_n)^{n-1}$ shall not cause exponent overflow in floating decimal.

over two hours; the tenth roots of unity takes about 20 minutes; a tenth degree polynomial with all real roots takes about 30 minutes, etc.

Appendix I - Operating Instructions

I. Servos and Tapes.

A. Servo 5 - Program tape.

B. Servo Δ - Blank for re-run purposes.

C. Servo 6 - Blank for edited output (optional)

D. Servo 7 - Polynomial (optional)

II. Breakpoints

A. Breakpoint No. 1.

1) Normal Path - Polynomial to be solved read in from servo 7 and servo 7 rewound. The tape must be prepared in the following manner: The first word to be typed must be 0000000000nn where n is the degree of the polynomial ($01 \leq nn \leq 20$). The next word must be the constant term of the polynomial, the next word the degree one term, degree two term, etc., up to the degree n term. The remainder of the block is immaterial. All coefficients must be in floating decimal composed form.

2) Force Transfer - Computer calls for the manual type in of the polynomial to be solved. In response to the type out "Ti DEGREE" type in 0000000000nn where n is the degree of the

polynomial ($01 \leq nn \leq 20$). There will then be a series of type outs of the form "TijjTERM" where $00 \leq jj \leq nn$. In response to each of these type outs, type in the j-th term (floating decimal composed) of the polynomial.

B. Breakpoint No. 2.

1) Normal Path - Final edited result will be written out at 20 pulses/inch on servo 6, servo 6 will be rewound, and the number of blocks written out will by typed out on SCT.

2) Force Transfer - Final edited result will be printed on the SCT (see below for typewriter instructions).

C. Breakpoint No. 9.

From time to time servo △ will be written on and then rewound. To re-run from that point (1) rewind servo 5 if necessary, (2) initial read servo 5, (3) force transfer on breakpoint No. 9.

III. Typewriter Instructions.

A. Always keep the SCT on normal.

B. For printing final edited results use narrow paper and keep both left and right hand margins as far over as possible. No tabs are necessary. There is a ∅ at the beginning of the editing and 3∅'s at the end.

To start the program initial read servo 5.

Appendix II -- Sample Problem

Polynomial

```
degree  00  term = +.100000000  (+01)
degree  01  term = +.000000000  (+00)
degree  02  term = +.000000000  (+00)
degree  03  term = +.000000000  (+00)
degree  04  term = +.000000000  (+00)
degree  05  term = +.000000000  (+00)
degree  06  term = +.000000000  (+00)
degree  07  term = +.000000000  (+00)
degree  08  term = +.000000000  (+00)
degree  09  term = -.200000000  (+01)
degree  10  term = +.000000000  (+00)
degree  11  term = +.000000000  (+00)
degree  12  term = +.000000000  (+00)
degree  13  term = +.000000000  (+00)
degree  14  term = +.000000000  (+00)
degree  15  term = +.000000000  (+00)
degree  16  term = +.000000000  (+00)
degree  17  term = +.000000000  (+00)
degree  18  term = +.100000000  (+01)
```

Roots

```
   real part          imag.part          error          mult.
-.939692616 (+00)  +.342020144 (+00)  +.842020144 (-07)   02
-.939692616 (+00)  -.342020144 (+00)  +.842020144 (-07)   02
-.500000004 (+00)  +.866025418 (+00)  +.243683202 (-06)   02
-.500000004 (+00)  -.866025418 (+00)  +.243683202 (-06)   02
+.173648179 (+00)  +.984807732 (+00)  +.566855781 (-07)   02
+.173648179 (+00)  -.984807732 (+00)  +.566855781 (-07)   02
+.766044438 (+00)  +.642787590 (+00)  +.379538043 (-06)   02
+.766044438 (+00)  -.642787590 (+00)  +.379538043 (-06)   02
+.999999996 (+00)  +.000000000 (+00)  +.400000000 (-07)   02
```

finished

Date Due

DEC 28 '56			
OCT 5 '58			
NOV 19 '65			
OCT 14 '66			
NOV 3 '66			

NYO-6482 cop. 1

NYO-6482 cop. 1

Isaacson

AUTHOR

A report on a modified

TITLE Routh method for locating roots of polynom.

NYO-6482 cop. 1

Isaacson.

A report on a modified Routh method for locating roots of polynomials.